AF/2123 2700

Please type a plus sign inside this box → [+]

PTO/SB/21 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/241,735
Filing Date	February 2, 1999
First Named Inventor	Hiroaki Kimura
Group Art Unit	2123
Examiner Name	F. Ferris
Total Number of Pages in This Submission: 120	Attorney Docket Number: 21776-00034-US

ENCLOSURES (*check all that apply*)

- [ ] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Assignment Papers *(for an Application)*
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
- [ ] After Allowance Communication to Group
- [ ] Appeal Communication to Board of Appeals and Interferences
- [X] Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- [ ] Proprietary Information
- [ ] Status Letter
- [X] Other Enclosure(s) *(please identify below)*

Add'l copies (2)

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual Name	CONNOLLY BOVE LODGE & HUTZ LLP Larry J. Hume - 44,163
Signature	Larry J Hume
Date	March 12, 2003

Docket No.: 21776-00034-US
(PATENT)

#13 Appeal
3-19-03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of: Conf. No. 9109

Hiroaki Kimura, et al.

Application No.: 09/241,735 Group Art Unit: 2123

Filed: February 2, 1999 Examiner: F. Ferris

For: APPARATUS FOR ANALYZING SOFTWARE AND METHOD OF THE SAME

SUBSTITUTE APPELLANT'S BRIEF

Commissioner for Patents
Attn: Board of Patent Appeals and Interferences
Washington, DC 20231

March 12, 2003

Dear Sir:

This substitute brief is in furtherance of the Notice of Appeal, filed in this case on August 8, 2002, Appellants' Brief filed December 30, 2002, the Notice of Non-Compliance with 37 CFR §1.192(c), mailed by the Examiner on February 19, 2003, and the telephone conversation with Supervisory Examiner Kevin Teska on February 28, 2003.

This Substitute Appeal Brief is filed to replace the Appellants' Brief previously filed on December 30, 2002, as required by the Examiner.

Although no fees are believed to be required for filing this brief, if any fees are required, authorization is hereby granted to charge CBLH Deposit Account No. 22-0185.

This brief is transmitted in triplicate.

This brief contains all required items under headings in the order required by 37 C.F.R. §1.192 and M.P.E.P. §1206, and also includes additional, non-required heading Roman numeral "X.", and Appendix B, structured to convey a better appreciation of the claims on appeal.

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is NS Solutions Corporation, Tokyo, Japan

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences which will directly affect or be directly affected by, or have any bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application: There are 37 claims pending.

B. Current Status of Claims

1. Claims canceled: none
2. Claims withdrawn from consideration but not canceled: none
3. Claims pending: 1-37
4. Claims allowed: none
5. Claims rejected: 1-37

C. Claims On Appeal: The claims on appeal are claims 1-37.

IV. STATUS OF AMENDMENTS

Applicants filed an Amendment in this case under 37 C.F.R. §1.111 on April 8, 2002, in response to the non-final Official Action dated November 7, 2001. The Examiner responded to the Amendment with a Final Rejection mailed May 8, 2002 In the Final Official Action, the Examiner indicated that Claims 1-37 were finally rejected.

Accordingly, the claims enclosed herein as Appendix A incorporate the amendments indicated in the paper filed by Applicant on April 8, 2002.

V. SUMMARY OF THE INVENTION

To set the stage for this appeal, and in light of the Examiner's comments and apparent misunderstanding of the novel and non-obvious aspects of the claimed invention, and to place the claims on appeal in their proper context for the Honorable Board, a background discussion of conventionally-available software analysis information is submitted as Appendix B to this Brief, in an effort to enable a better appreciation of the claims on appeal.

As a further preliminary matter, Appellants note that the claims, as originally filed, may be relied upon to provide supporting disclosure, as the claims as filed in the original Specification are part of the disclosure.[1]

A. Introductory Remarks

This optional, introductory subsection is provided as an aid to the Examiner and the Honorable Board in understanding the claimed invention, without specific reference to Specification page and line numbers. Specific reference to page and line numbers in the Specification which provide enabling disclosure of the claimed invention, is provided in subsection "B" of this Summary of the Invention Section, as required by the MPEP and 37 C.F.R. §1.192.

The disclosed and claimed invention relates generally to a software analysis apparatus, a software analysis method, and a computer-readable medium recording a computer program for

---

[1] *See* MPEP §2163.06(III).

making a computer implement functions related to analyzing a computer program to help the user easily understand the contents of the computer program, and the complex interactions within the program. More specifically, the disclosed and claimed invention relates to a software analysis apparatus, method, and computer-readable medium for analyzing conventionally obtained program analysis information in a new way.

Program analysis information generated by conventional program analysis information generation means is stored in a data recording medium, and is preferably stored, in one claimed embodiment, in the form of an object-oriented database. Thus, program analysis information from a large scale program may be safely and reliably obtained and retrieved without experiencing the conventional problems of a memory shortfall in the middle of analysis, even when the memory capacity mounted on the computer or analysis workstation is limited.

Further, even when the analysis process is interrupted due to some computer or operator-related problem, e.g., memory shortage, or electrical power outage, information stored in the data recording medium, at least to the point of occurrence of the problem, is safely protected. Hence, in contrast with conventional software analysis approaches, the analysis process is able to be resumed from the point where the process was interrupted, thus avoiding repetitive processing that would be required to reinitiate processing from the beginning. This greatly improves analysis efficiency, and reduces operator effort and time required to complete the analysis.

When an analysis process is to be interactively accomplished after generation of conventional analysis information, and since analysis information already stored in the database can be directly used, the time required for generating that analysis information can be omitted, and the wait time of the operator can be reduced compared to a conventional system which generates analysis information from the beginning, every time new analysis process is started due to a previously occurring technical problem which cut short the original analysis before processing had completed.

The disclosed and claimed invention has been developed in consideration of the discussion concerning handling of conventionally-obtained program analysis information in the Background Section of the originally-filed application, and as further discussed in APPENDIX B of this Brief.

B. Disclosure Relating to the Claimed Embodiments

In one embodiment of the invention claimed in independent claim 1 (Group I), a software analysis apparatus includes program analysis information generation means for automatically generating program analysis information required for analyzing a computer program (*See* Specification at p. 5, lines 9-12, p. 7, line 24, through p. 8 line 5, and original claim 1, lines 2-4); program analysis information storage means (*See* Specification at p. 5, lines 12-15, and p. 8, line 6) for classifying the program analysis information generated by said program analysis information generation means in an arbitrary unit or at an arbitrary timing (*See* Specification at p. 16, lines 17-22, e.g., classified in accordance with Fig. 2, in an exemplary embodiment, as a particular type of data), and sequentially storing the program analysis information in a predetermined data recording medium (*See* Specification at p. 5, lines 13-17, p. 8, lines 6-18, and original claim 1, lines 5-9); and program analysis means for executing program analysis by reading out the program analysis information from the data recording medium (*See* Specification at p. 5, lines 17-20, p. 8, lines 20-24, p. 18, line 14 through p. 22, line 28, and original claim 1, lines 11-13). In addition, see Figs. 1, 3, and 4 of the Drawings.

In this and other embodiments discussed below, and as discussed in the Specification, the type of program analysis information which is classified, stored, and analyzed may include, for example, the following types of conventional analysis information: Syntactic analysis tree (including a symbol table), call graph, flow graph, data flow information, program dependence graph, module I/O information, metrics information, redundancy information, maintenance document information, as shown in Fig. 2. Of this type of information, the syntactic analysis tree, call graph, flow graph, data flow information, program dependence graph, module I/O information, for example, are generated by the program analysis information generation unit 11 shown in Fig. 1, while the metrics information, redundancy information, maintenance document information, for example, are generated by a program analysis unit 20_i shown in Fig. 3, that analyzes a program by a batch process. (*See* Specification at p. 9, line 20 through p. 10, line 6; further details of these conventional types of program analysis information are found in the Specification at p. 10, line 7, through p. 14, line 20).

In an embodiment of the invention claimed in independent claim 13 (Group II), a software analysis method includes automatically generating program analysis information

required for analyzing a computer program (*See* Specification at p. 5, lines 23-25, p. 7, line 24, through p. 8, line 5, and original claim 13, lines 2-4); classifying the program analysis information in an arbitrary unit or at an arbitrary timing (*See* Specification at p. 16, lines 17-22, and p. 5, lines 16-17); sequentially storing the program analysis information in a predetermined data recording medium (*See* Specification at p. 5, lines 13-16, p. 8, lines 6-18, and original claim 13, lines 5-9); and executing program analysis by reading out the program analysis information from the data recording medium (*See* Specification at p. 6, lines 2-5, p. 8, lines 20-24, p. 18, line 14 through p. 22, line 28, and original claim 13, lines 11-13). In addition, see Figs. 1, 3, and 4 of the Drawings.

In an embodiment of the invention claimed in independent claim 24 (Group III), a computer readable recording medium is disclosed which has a computer program recorded thereon for making a computer implement various functions, including a program analysis information generation function of automatically generating program analysis information required for analyzing a computer program (*See* Specification at p. 6, lines 6-11, p. 7, line 24, through p. 8, line 5, and original claim 24, lines 3-5); a program analysis information classification function of classifying the program analysis information generated by the program analysis information generation function in an arbitrary unit or at an arbitrary timing (*See* Specification at p. 16, lines 17-22); a program analysis information storage function of sequentially storing the classified program analysis information generated by the program analysis information classification function in a predetermined data recording medium (*See* Specification at p. 5, lines 12-17, p. 8, lines 6-18, and original claim 1, lines 6-10); and a program analysis function of executing program analysis by reading out the classified program analysis information from the data recording medium (*See* Specification at p. 5, lines 17-20, p. 8, lines 20-24, p. 18, line 14 through p. 22, line 28, and original claim 24, lines 12-14). In addition, see Figs. 1, 3, and 4 of the Drawings.

In an aspect of the embodiment of the invention variously claimed in dependent claims 7, 18, and 29 (Group IV), depending from independent claims 1, 13, and 24, respectively, the program analysis information storage means of claim 7 stores the program analysis information in the data recording medium as a database (*See* Specification at p. 5, lines 17-20, p. 8, lines 6-10, and original claim 7, lines 2-4) or, alternatively, in claims 18 and 29, a step or function of storing the program analysis information in the data recording medium as a database is

performed (*See* Specification at p. 5, lines 17-20, p. 8, lines 6-10, original claim 18, lines 2-4, and original claim 29, lines 2-4).

In an embodiment of the invention claimed in independent claim 11 (Group IV), a software analysis apparatus for generating program analysis information required for analyzing a computer program includes means for hierarchically registering the generated program analysis information in a database in units of analysis objectives (*See* Specification at p. 8, lines 6-8, p. 14, line 21 through p. 15, line 4, p. 16, lines 17-24, and original claim 11, lines 1-5); means for implementing analysis of the hierarchically registered program analysis information by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program (*See* Specification at p. 8, lines 19-24, p. 15, line 28 through p. 16, line 6, original claim 11, lines 5-9, and Fig. 2).

In an embodiment of the invention claimed in independent claim 22 (Group IV), a software analysis method for generating program analysis information required for analyzing a computer program includes hierarchically registering the generated program analysis information in a database in units of analysis objectives (*See* Specification at p. 8, lines 6-8, p. 14, line 21 through p. 15, line 4, p. 16, lines 17-24, and original claim 22, lines 3-5), and implementing analysis by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program (*See* Specification at p. 8, lines 19-24, p. 15, line 28 through p. 16, line 6, original claim 22, lines 5-9, and Fig. 2).

In an embodiment of the invention claimed in independent claim 33 (Group IV), a computer readable storage medium is disclosed which records a program for making a computer implement the functions of generating program analysis information required for analyzing a computer program (*See* Specification at p. 5, lines 23-25, p. 7, line 24 through p. 8, line 5, p. 24, line 18 through p. 25, line 2, and original claim 33, lines 1-4), hierarchically registering the generated program analysis information in a database in units of analysis objectives (*See* Specification at p. 8, lines 6-8, p. 14, line 21 through p. 15, line 4, p. 16, lines 17-24, and original claim 33, lines 4-6), and implementing analysis by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon

analyzing the computer program (*See* Specification at p. 8, lines 19-24, p. 15, line 28 through p. 16, line 6, and original claim 33, lines 7-10, and Fig. 2).

With respect to the various apparatus claim element limitations relying upon means plus function and method claim limitations, the Specification, at least at p. 9, lines 9-14, discloses that the various elements described above may be implemented in a computer having a CPU, ROM, and RAM, for example.

In the interests of brevity, and as suggested by Supervisory Examiner Teska, support for at least the independent claims of each claim Group I-IV has been provided above.

VI. ISSUES

A. Has the Examiner established either that the Specification lacks enabling support or written description support for conventional aspects of claims 1-37 under 35 U.S.C. §112, first paragraph, given the general knowledge and skill available to a person having skill in the art, in light of known, conventional approaches at the time of Appellants' invention, especially given the Examiner's admission that such aspects were actually conventional and known at that time, and given the complete lack of reference in the rejection to any specific limitation in the claims which is allegedly not enabled or described?

B. Has the Examiner established that every recited feature of claims 1-6, 8-10, and 35 is disclosed by Wygodny et al. (US 6,282,701), and is therefore anticipated under 35 U.S.C. §102(e)?

C. Has the Examiner established that every recited feature of claims 13-17, 19-21, and 36 is disclosed by Wygodny et al. (US 6,282,701), and is therefore anticipated under 35 U.S.C. §102(e)?

D. Has the Examiner established that every recited feature of claims 24-28, 30-32, and 37 is disclosed by Wygodny et al. (US 6,282,701), and is therefore anticipated under 35 U.S.C. §102(e)?

E. Has the Examiner established that every recited feature of claims 7, 11, 12, 18, 22, 23, 29, 33, and 34 is disclosed by Wygodny et al. (US 6,282,701), and is therefore anticipated under 35 U.S.C. §102(e)?

VII. GROUPING OF CLAIMS

For purposes of this appeal brief only, and without conceding the teachings of any prior art reference, the claims have been grouped as indicated below:

Group	Claims
I	1-6, 8-10, and 35
II	13-17, 19-21, and 36
III	24-28, 30-32, and 37
IV	7, 11, 12, 18, 22, 23, 29, 33, and 34

In Section IX below, Applicant has included separate arguments supporting the separate patentability of each claim group as required by M.P.E.P. §1206.

VIII. ARGUMENTS

A. The Examiner has not established that the Specification lacks either enabling support or written description support for conventional aspects of claims 1-37 under 35 U.S.C. §112, first paragraph, given the Examiner's admission that such aspects were conventional and known at that time, and given the complete lack of reference in the rejection to any allegation that any specific limitation in the claims is not enabled or described.

By way of introduction, the rejection under §112, first paragraph is deficient, in that it does not clearly state whether the rejection is for lack of written description, or for lack of enablement.

The explicit statement of the rejection is further deficient in that it does not identify ***any*** limitation recited in ***any*** claim on appeal which is asserted as lacking enabling and/or written description support. It is not enough to merely object to the Specification as a basis for rejection under 35 U.S.C. §112, first paragraph. An explicit statement of the rejection must be provided which asserts, with specificity, the claim limitation(s) which are allegedly not enabled by the instant Specification.

The Examiner merely rejected claims 1-37 under 35 U.S.C. §112, first paragraph, "for the reasons set forth in the objections to the Specification."[2] This rejection is rooted in the non-appealable objection to the Specification as not containing translations of Appellants' background Japanese references.

Appellants submit that the Examiner's objection to the Specification is unreasonable as it relates to the §112, first paragraph rejection, because the foreign applications mentioned briefly in the Background section are merely indicative of the conventional state of the art, and are not necessary to a proper understanding of the claimed invention.

Although objections are typically not appealable and are usually only addressable by Petition to the Commissioner, as the Board of Appeals and Interferences has held, "[t]hat part of the Examiner's "objection" which centers on description, enablement and best mode concerns the correspondence of the specification to the statutory requirements set forth in 35 U.S.C. §112 and is within the jurisdiction of this Board."[3]

Appellants point out that "essential material" is defined as that which is necessary to (1) describe the claimed invention, (2) provide an enabling disclosure of the claimed invention, or (3) describe the best mode (35 U.S.C. 112). In any application which is to issue as a U.S. patent, essential material may not be incorporated by reference to (1) patents or applications published by foreign countries or a regional patent office, (2) non-patent publications, (3) a U.S. patent or application which itself incorporates "essential material" by reference, or (4) a foreign application.

However, ***nonessential*** subject matter may be incorporated by reference to (1) patents or applications published by the United States or foreign countries or regional patent offices, (2) prior filed, commonly owned U.S. applications, or (3) non-patent publications.[4] Nonessential subject matter is subject matter referred to for purposes of indicating the background of the invention, or illustrating the state of the art. Appellants submit that this is the case in this

---

[2] *See* Final Official Action, p. 3, paragraph 4.
[3] *Ex parte* C, 27 USPQ 2d 1492, 1494 (B.P.A.I. 1993).
[4] *See* MPEP §608.01.

Appeal.

The Examiner correctly observes that the specification includes a reference to Japanese patent Applications, which he asserts are part of the claimed invention's description. The Examiner incorrectly concludes, however, that these references, mentioned in passing in the "Description of Related Art" section of the Specification (*See* Specification at p. 1, line 27 through p. 2, line 2), constitute "essential matter".

The Examiner objected that Appellants' background Japanese references (i.e., JP 9-32415 and JP 9-32452) must be translated into English, and incorporated into the disclosure. In addition, the Examiner indicated that any translation must be accompanied by an affidavit or declaration executed by the Applicant or Applicant's Representative stating that the material added to the Specification consists of the same material incorporated by reference in the application.

Appellants submit that none of their cited Japanese applications, mentioned in passing in the Background Section, are "essential material" which must be incorporated bodily into the Specification by translation and amendment. These references have been mentioned only to apprise the reader of some of the conventional program analysis approaches available prior to this application (See Specification at p. 2, line 27 -28). Appellants' cited Japanese applications are also mentioned only briefly in the Disclosure with respect to indicating known methods of detection of "data flow anomaly" (See Specification at p. 12, lines 25-28). Appellants submit that this aspect of conventional techniques should also not be considered "essential material" for the purposes of this application.

These references have been referred to in the Specification only in an attempt to indicate, in passing, some of the available conventional program analysis techniques or tools, and are submitted by Appellants as being "non-essential".

The Examiner goes on to incorrectly assert that Appellants have claimed an apparatus and method for "program analysis" as a basis for all independent claims, and have not claimed

the conventional types of information discussed below, and in Appendix B to this Brief.[5]

Appellants point out that independent claims 1 and 11 claim a ***software analysis*** apparatus; independent claims 13 and 22 claim a ***software analysis*** method; and independent claims 24 and 33 claim a ***computer readable recording medium*** recording a computer program for making a computer implement various unique and non-obvious functions relating to processing of conventionally generated program analysis information.

Appellants reiterate that the thrust of Appellants' invention is not the program analysis process, *per se*, but rather how information generated by conventional analysis processes is stored, used, and analyzed in a more efficient manner. Appellants submit that a person having skill in the art would know the various conventional "analysis processes" which could benefit from Applicants' invention, particularly in light of the Specification, and in further light of the knowledge and skill available to a person having skill in the art.

In fact, the Examiner admits "...that processes relating to source code, syntactic analysis tree, symbol table, call graph, flow graph, and data flow information are obvious and well known...."[6] Appellants reiterate that generation of this type of information, by itself, is not the invention; rather the claimed invention is directed to an apparatus and method for managing such conventionally generated information in a novel and non-obvious manner.

In either the case of a rejection for lack of written description, or for lack of enabling disclosure, the Examiner has not established that the Specification lacks enabling written support for the admitted (by both the Examiner and Appellants) conventional limitations claimed in claims 1-37. Further, in light of the case law cited in the subsections of this Brief below, conventional features are not required to be described in detail in the Specification.

Although Appellants submit that they should not be required to endure further expense and delay in prosecuting this Appeal, in the event that the Honorable Board believes that such conventional, background material objected to, in its absence, by the Examiner, is, indeed,

---

[5] *See* Final Official Action, p. 7, paragraph 6 through p. 8, line 2.
[6] *See* Final Official Action, p. 7, paragraph 6, lines 11-13.

"essential material" necessary to overcome the §112, first paragraph rejections, Appellants will arrange to translate the Japanese references, and submit an amendment which bodily incorporates the translated references into the Background section of the Disclosure.

Therefore, reversal of the §112, first paragraph rejections by the Honorable Board is respectfully requested.

1. Enablement

If the Examiner intended that the §112, first paragraph rejections are for lack of enabling disclosure, Appellants point out that the Federal Circuit has held,

> "[w]hen the challenged subject matter is a computer program that implements a claimed device or method, enablement is determined from the viewpoint of a skilled programmer using the knowledge and skill with which such a person is charged. The amount of disclosure that will enable practice of an invention that utilizes a computer program may vary according to the nature of the invention, the role of the program in carrying it out, and the complexity of the contemplated programming, all from the viewpoint of the skilled programmer.[7]

Appellants submit that it is known to those with skill in the art that metrics information, redundancy information and maintenance document information are generated on the basis of program analysis information, such as source code, syntactic analysis tree, symbol table, call graph, flow graph, data flow information, program dependence graph, and module I/O information, and that persons with skill in the art would also know how to generate such conventional program analysis information without undue experimentation. "The test of enablement is whether a person of ordinary skill in the relevant art, using his or her knowledge and the patent disclosure, could make and use the invention without undue experimentation."[8]

"Section 112 of 35 U.S.C. does not require that all of the elements necessary to make the invention work must be described in the specification. Section 112 rather requires that the specification contain a description of a claimed invention in such terms that will enable one

---

[7] *Northern Telecom Inc. v. Datapoint Corp.*, 15 USPQ2d 1321, 1329 (Fed. Cir. 1990).
[8] *Williams Service Group Inc. v. O.B. Cannon & Son Inc.*, 33 USPQ2d 1705, 1723 (Pa. 1994).

skilled in the art to make and use the invention."[9] Appellants submit that such is the case with the instant Specification.

Further, although software program metrics information, redundancy information, and maintenance document information relating to a software program may be conventionally generated as a result of a batch process, the specific methods actually used to generate such conventionally generated information are submitted as not being essential to an understanding of the disclosed and claimed invention, except to the extent necessary to provide a general background and contextual understanding of the area of computer and software technology impacted by Appellants' invention.

Further, to the extent that the enablement rejection of the claims is related to the Examiner's objection to the Specification with respect to his assertion that incorporation of non-essential, background material found in Japanese references is required, Appellants point out that case law supports the proposition that mere reference to another application, patent, or publication is not an incorporation of anything therein into the application containing such reference, for the purpose of the disclosure required by 35 U.S.C. 112, first paragraph.[10] The Japanese references are mentioned in passing as merely providing background information, and not essential material.

Although the disclosed and claimed apparatus, method, and computer-readable medium are submitted as being novel and non-obvious, ***the actual program analysis information which may be processed by the invention is submitted as being known in the art***, and may include information obtained from conventional software program analysis techniques such as, for example, graphically displayed call graph and flow graph, program influence range analysis, program structure analysis, and program data flow anomaly analysis, which are admitted by the Examiner as being known.

Reversal of the Examiner's rejection of claims 1-37 under 35 U.S.C. §112, first paragraph, is therefore requested.

---

[9] *Valmont Industries, Inc. v. Reinke Manufacturing Co.*, 14 USPQ2d 1374 (Neb. 1990).
[10] *In re de Seversky*, 474 F.2d 671, 177 USPQ 144 (CCPA 1973).

2. Written Description

Dealing now with the possibility that the §112, first paragraph rejections are based upon a purported lack of written description support, Appellants point out that support for the pending claims may be found, at least initially, for example, in the originally presented claims 1-34. Appellants point out that the claims as filed in the original Specification are part of the disclosure.[11]

The Examiner bases his enablement rejection on various references in the Specification to "program analysis" and "analysis process", which in his estimation, are a critical part of the invention, and which are not disclosed in explicit detail. Appellants again reiterate that these conventionally available techniques are not required to be disclosed in the Specification, as the invention is not drawn to these processes, *per se*, but is directed to storage, handling, and post-processing of the program analysis information which is generated by known techniques.

With respect to written description aspects of the §112, first paragraph rejections, the Examiner asserts that no algorithms are given and no analytical process is described, and that the specification makes reference to the claimed invention analyzing a computer program, and automatically generating program analysis information, but goes on to say that the Specification does not specifically explain the process.

"The burden of showing that the claimed invention is not described in the application rests on the PTO in the first instance, and it is up to the PTO to give reasons why a description not in *ipsis verbis* is insufficient."[12]

Appellants submit that, since each of these conventional processes used in various ways as precursor information for the inventive claimed apparatus and method are generally known, the instant specification is not required to discuss these processes in detail. "An applicant (patentee) may begin at a point where his invention begins and describe what he has made that is

---

[11] *See* MPEP §2163.06(III).
[12] *In re Edwards, Rice, and Soulen*, 196 USPQ 465, 469 (C.C.P.A. 1978).

new, and what it replaces of the old. That which is common and well known is as if it were written out in the application (patent) and delineated in the drawings."[13]

Further, "[u]nder 35 U.S.C. §112, a specification need not teach that which is obvious to those in the art."[14] "A patent need not teach, and preferable omits, what is well know in the art."[15] Finally, "[t]he disclosure of an application embraces not only what is expressly set forth in words and drawings, but what would be understood by persons skilled in the art. Those features that are well known are as if they were written out in the patent."[16]

Notwithstanding the above-cited case law, the instant specification provides description of the conventional processes executed by the program analysis means, such as call graph and flow graph which are graphically displayed, influence range analysis, structure analysis and data flow anomaly analysis (See App. B of this Brief, and Specification at p. 9, line 20 through p. 14, line 20).

Further, the Examiner asserts that processes such as "metrics information", "redundancy information", and "maintenance document information" do not have an equivalent specific meaning in the art, and are not sufficiently defined by the Specification. Appellants traverse this assertion, because these terms are submitted as being adequately defined in Appellants' Specification.

Appellants, in their Specification, have provided definitions of each of these terms, as discussed in APPENDIX B of this Appeal Brief, and as repeated below with reference to the Specification.

For example, Appellants submit that "Metrics information" is known to be a measure of software quality which indicate the complexity, understandability, testability, description and intricacy of code, and which pertains to numeration indices of software. (See Specification, as previously amended, at p. 13, lines 15-27).

---

[13] *Webster Loom Co. v. Higgins*, 105 U.S. 580, 586 (1882).
[14] *In re Sureau, Kremer, and Dupre*, 153 USPQ 66, 70 (C.C.P.A. 1967).
[15] *Spectra-Physics Inc. v. Coherent Inc.*, 3 USPQ2d 1737, 1743 (Fed. Cir. 1987).
[16] *Ex parte Wolters and Kuypers*, 214 USPQ 735 (PTO Bd. App. 1979).

"Redundancy information" pertains to a redundant sentence that does not influence output in a single procedure. (See Specification, at p. 14, lines 1-8).

"Maintenance document information" refers to a document group used upon maintaining a program. (See Specification, at p. 14, lines 9-20).

Accordingly, Appellants submit that the terms above, which were mentioned by the Examiner as not having equivalent specific meaning in the art, are adequately defined by the instant Specification, particularly considering the knowledge of a person having skill in the art with respect to the known program analysis tools and techniques.

Therefore, Appellants submit that a person having skill in the art can fully understand and appreciate the novel and non-obvious features of the recited invention from the description contained in the present specification.

Reversal of the Examiner's rejection of claims 1-37 under 35 U.S.C. §112, first paragraph, is therefore requested.

B. The Examiner has not established that every recited feature of claims 1-6, 8-10, and 35 in Group I is disclosed by Wygodny et al. (US 6,282,701), and has therefore not established anticipation under 35 U.S.C. §102(e).

Applicants note that anticipation requires the disclosure, in a prior art reference, of each and every limitation as set forth in the claims.[17] There must be no difference between the claimed invention and reference disclosure for an anticipation rejection under 35 U.S.C. §102.[18] To properly anticipate a claim, the reference must teach every element of the claim.[19] "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference".[20] "The identical invention must be shown in as

---

[17] *Titanium Metals Corp. v. Banner*, 227 USPQ 773 (Fed. Cir. 1985).
[18] *Scripps Clinic and Research Foundation v. Genentech, Inc.*, 18 USPQ2d 1001 (Fed. Cir. 1991).
[19] *See* MPEP § 2131.
[20] *Verdegaal Bros. v. Union Oil Co. of Calif.*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

complete detail as is contained in the ...claim."[21] In determining anticipation, no claim limitation may be ignored.[22]

In view of the foregoing authority, Wygodny et al. at least fails to anticipate independent claim 1, and also fails to anticipate dependent claims 2-6, 8-10, and 35, depending from claim 1.

With reference to the general differences in the claimed invention, and with particular respect to the recited distinctions of pending claims 1-6, 8-10, and 35 of the claimed invention over Wygodny et al., and considering the Examiner's comment that Wygodny et al. disclose a program analyzer method and apparatus having a GUI (Graphical User Interface) which collects trace information for use in analyzing a computer program, ***Applicants point out that, contrary to the Examiner's assertion, "trace", as used in Wygodny et al., and "analysis information" in the present invention are technically different concepts.***

The disclosed and claimed invention uses program analysis information obtained from analysis of a non-executing software program. The types of conventional program analysis information are ***known as being obtained from a non-executing software program***. Under the applicable case law for anticipation, it is not enough that the Examiner find a reference which discloses something in a related technical field, i.e., computer software analysis. The Examiner must find a single reference which discloses every claimed limitation.

Wygodny et al. facilitates the process of tracing the program execution paths as the program executes, and relates to the process of monitoring and analyzing the execution of computer programs during the debugging process. In Wygodny et al., the tracing is performed without requiring modifications to the executable modules or source code, and the trace data is collected according to instructions in a trace control dataset. Wygodny et al. propose a software system that facilitates the process of identifying and isolating bugs within a client program by allowing a developer to trace the execution paths of the client.

---

[21] *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).
[22] *Pac-Tex, Inc. v. Amerace Corp.*, 14 USPQ2d 187 (Fed. Cir. 1990).

Generally, a debugger examines how variable values change as a program executes. The program analysis information of the disclosed and claimed invention, such as syntactic analysis tree, call graph, and flow graph, is statically obtained from source code, ***but is not generated during execution of the program***.

The trace data as in Wygodny indicates values of executed functions and variable values when the program is executed. The volume of trace data may become large if the time required for executing programs is long, or if a large number of instructions are executed. Further, the trace data of Wygodny et al. is trace information generated by a debugger, and generated concurrent with the execution of the program.

A "trace" is conventionally defined as a debugging aid that chronicles the actions and results of individual steps in a program. A trace takes a user's program and places it under control of a special routine which monitors the progress of the program. Continuous execution of the user's program is replaced by a process whereby the trace program intercedes between steps of the user's program, displaying a variety of material before permitting execution of the next step. The contents of most types of traces are characterized by such items as a copy of the instruction, its location, and operand and register values before and after execution. Some trace facilities are concerned with the sequence of events in a program, as well as with the history of various data items, so that indications may be included as to whether certain branches have been followed, or information about cyclic processes, as the processes are being actually executed.[23]

As for the specific deficiencies of the applied art with respect to the claimed invention, Wygodny et al. does not disclose a software analysis apparatus which includes, among other features, "...program analysis information generation means for automatically generating program analysis information required for analyzing a computer program; program analysis information storage means for classifying the program analysis information generated by said program analysis information generation means in an arbitrary unit or at an arbitrary timing, and sequentially storing the program analysis information in a predetermined data recording medium...", as recited in independent claim 1.

---

[23] Encyclopedia of Computer Science, p. 1427, Van Nostrand Reinhold Co., New York, 1976.

Wygodny et al. not only does not disclose generation of program analysis information, as that term would be understood by a reading of Appellants' disclosure, but Wygodny et al. also does not disclose program analysis information storage means for storing and classifying the program analysis information.

In the Examiner's response to previously submitted arguments, the Examiner asserts, without offering any evidence for the record, that "merely gathering and storing information to processes related to [known information types]...does not constitute the basis for that which is unique or novel",[24] and further that "it would have been obvious to classify syntactic data as such, or symbol data as such...but this would not be unique or novel."[25]

There is no evidence in the record to support these allegations of lack of novelty or of obviousness. Further, the rejections on appeal are for anticipation under 35 U.S.C. §102, and not for unpatentability under §103.

Accordingly, as the applied art does not disclose all the claimed features, the Examiner has not met his evidentiary burden for anticipation, and reversal of the rejection of independent claim 1, and dependent claims 2-6, 8-10, and 35 by the Honorable Board is requested.

C. The Examiner has not established that every recited feature of claims 13-17, 19-21, and 36 in Group II is disclosed by Wygodny et al. (US 6,282,701), and has therefore not established anticipation under 35 U.S.C. §102(e).

The citations to relevant case law on anticipation are provided in paragraph B above.

Wygodny et al. does not disclose a software analysis method, which includes, among other features, "...classifying the program analysis information in an arbitrary unit or at an arbitrary timing; sequentially storing the program analysis information in a predetermined data recording medium; and executing program analysis by reading out the program analysis information from said data recording medium", as recited in independent claim 13.

---

[24] *See* Final Official Action, p. 9, lines 13-19.
[25] *See* Final Official Action, p. 10, lines 9-10.

The Examiner admits, in his response to arguments cited above, that Wygodny et al. is deficient with respect to disclosing classifying program analysis information, but asserts, without offering any evidence, that it would be "obvious" to classify data, and that it is not novel to store program analysis information, as discussed above.

There is no evidence in the record to support these allegations of lack of novelty or of obviousness. Further, Appellants repeat that the rejections on appeal are for anticipation under 35 U.S.C. §102, and not for unpatentability under §103.

Accordingly, as the applied art does not disclose all the claimed features, the Examiner has not met his evidentiary burden for anticipation, and reversal of the rejection of independent claim 13, and dependent claims 14-17, 19-21, and 36 by the Honorable Board is requested.

D. The Examiner has not established that every recited feature of claims 24-28, 30-32, and 37 in Group III is disclosed by Wygodny et al. (US 6,282,701), and has therefore not established anticipation under 35 U.S.C. §102(e).

The citations to relevant case law on anticipation are provided in paragraph B above.

Wygodny et al. does not disclose a computer readable recording medium recording a computer program for making a computer implement, among other functions, "...a program analysis information classification function of ***classifying*** the program analysis information generated by the program analysis information generation function in an arbitrary unit or at an arbitrary timing; a program analysis information storage function of sequentially ***storing*** the classified program analysis information generated by the program analysis information classification function in a predetermined data recording medium; and a program analysis function of executing program analysis by ***reading*** out the classified program analysis information from said data recording medium", as recited in independent claim 24.

The Examiner admits, in his response to arguments cited above, that Wygodny et al. is deficient with respect to disclosing classifying program analysis information, but asserts, without offering any evidence, that it would be "obvious" to classify data, and that it is not novel to store program analysis information, as discussed above.

There is no evidence in the record to support these allegations of lack of novelty or of obviousness. Further, Appellants again reiterate that the rejections on appeal are for anticipation under 35 U.S.C. §102, and not for unpatentability under §103.

Accordingly, as the applied art does not disclose all the claimed features, the Examiner has not met his evidentiary burden for anticipation, and reversal of the rejection of independent claim 24, and dependent claims 25-28, 30-32, and 37 by the Honorable Board is requested.

E. The Examiner has not established that every recited feature of claims 7, 11, 12, 18, 22, 23, 29, 33, and 34 in Group IV is disclosed by Wygodny et al. (US 6,282,701), and has therefore not established anticipation under 35 U.S.C. §102(e).

The citations to relevant case law on anticipation are provided in paragraph B above.

A common feature of the claims of Group IV is that program analysis information is stored in a data recording medium as a database and, in at least one dependent claim (34), in an object-oriented database.

Wygodny et al. does not disclose a software analysis apparatus which includes, among other features, "...program analysis information storage means for classifying the program analysis information generated by said program analysis information generation means in an arbitrary unit or at an arbitrary timing, and sequentially storing the program analysis information in a predetermined data recording medium... wherein said program analysis information storage means stores the program analysis information in said data recording medium as a database", as recited in dependent claim 7.

In the final rejection, the Examiner does not address any specific, asserted disclosure in anticipation of dependent claim 7.

Wygodny et al. does not disclose a software analysis apparatus for generating program analysis information required for analyzing a computer program, which includes, among other features, "...means for hierarchically registering the generated program analysis information in a database in units of analysis objectives; [and] means for implementing analysis of the hierarchically registered program analysis information by reading out the program analysis

information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program", as recited in independent claim 11.

The Examiner does not address any asserted disclosure of dependent claim 11, other than to erroneously assert that Figs. 3A, 3B, 5-7, and 13-14 disclose storing analyzed data in a database.

What these figures actually disclose is as follows: Fig. 3A - an illustration of a typical main frame window provided by the system's trace analyzer module; Fig. 3B – an illustration of a typical main frame window showing multiple threads; Fig. 5 – a trace option window that allows a developer to select the functions to be traced and the information to be collected during the trace; Fig. 6 – a file page window that provides a hierarchical tree of trace objects listed according to hierarchical level; Fig. 7 – a class page window that provides a hierarchical tree of trace objects sorted by class; Fig. 13 – a flowchart which illustrates the process of attaching to (hooking) a running process; and Fig. 14 – a flowchart which illustrates the process of loading an executable file and attaching to (hooking) the program.

None of these figures in Wygodny et al. offered by the Examiner disclose a database or database structure.

In particular, Wygodny et al. does not disclose a software analysis method, which includes, among other features, "...classifying the program analysis information in an arbitrary unit or at an arbitrary timing; sequentially storing the program analysis information in a predetermined data recording medium; and executing program analysis by reading out the program analysis information from said data recording medium...wherein the program analysis information storage step includes the step of storing the program analysis information in said data recording medium as a database", as recited in dependent claim 18.

The Examiner does not address any specific, asserted disclosure in anticipation of dependent claim 18.

Wygodny et al. does not disclose a software analysis method for generating program analysis information required for analyzing a computer program, which includes, among other

features, "...hierarchically registering the generated program analysis information in a database in units of analysis objectives...", as recited in independent claim 22.

The Examiner does not address any specific, asserted disclosure in anticipation of independent claim 22 which relates to use of a database.

Wygodny et al. does not disclose a computer readable recording medium recording a computer program for making a computer implement, among other functions, "...a program analysis information classification function of classifying the program analysis information generated by the program analysis information generation function in an arbitrary unit or at an arbitrary timing; a program analysis information storage function of sequentially storing the classified program analysis information generated by the program analysis information classification function in a predetermined data recording medium...wherein the program analysis information storage function stores the program analysis information in said data recording medium as a database", as recited in dependent claim 29.

The Examiner does not address any specific, asserted disclosure in anticipation of dependent claim 29.

Finally, Wygodny et al. does not disclose a computer readable storage medium recording a program for making a computer implement, among other functions, "...hierarchically registering the generated program analysis information in a database in units of analysis objectives...", as recited in independent claim 33.

The Examiner does not address any specific, asserted disclosure in anticipation of independent claim 33, other than to assert, apparently with respect to dependent claim 34, without offering any evidence, that "[o]bject-oriented database techniques are old and well known methods of database construction."[26]

---

[26] *See* Final Official Action at p. 6, lines 18-19.

Appellants point out that there is no evidence in the record to support these allegations of "well-known" technology, and Wygodny et al. certainly does not disclose this dependent claim feature of claim 34.

Appellants again reiterate that the rejections on appeal are for anticipation under 35 U.S.C. §102, and not for unpatentability under §103. There has been no indication of any art of record, which is properly combinable with Wygodny et al. in an unpatentability rejection, and which teaches or suggests all the recited features.

In terms of reliance upon common sense or ("well known") common knowledge, it is worth noting that, in *Zurko*, the Federal Circuit has reiterated to the Board of Appeals the following:

> ...the deficiencies of the cited references cannot be remedied by the Board's general conclusions about what is "basic knowledge" or "common sense" to one of ordinary skill in the art...the Board contended that even if the cited UNIX and FILER2 references did not disclose a trusted path, "it is basic knowledge that communication in trusted environments is performed over trusted paths" and, moreover, verifying the trusted command in UNIX over a trusted path is "nothing more than good common sense." *Ex parte Zurko*, slip op. at 8.
>
> We cannot accept these findings by the Board. This assessment of basic knowledge and common sense was not based on any evidence in the record and, therefore, lacks substantial evidence support. As an administrative tribunal, the Board clearly has expertise in the subject matter over which it exercises jurisdiction. This expertise may provide sufficient support for conclusions as to peripheral issues. With respect to core factual findings in a determination of patentability, however, the Board cannot simply reach conclusions based on its own understanding or experience -- or on its assessment of what would be basic knowledge or common sense.
>
> Rather, the Board must point to some concrete evidence in the record in support of these findings. To hold otherwise would render the process of appellate review for substantial evidence on the record a meaningless exercise. *Baltimore & Ohio R.R. Co. v. Aderdeen & Rockfish R.R. Co.*, 393 U.S. 87, 91-92 (1968) (rejecting a determination of the Interstate Commerce Commission with no support in the record, noting that if the Court were to conclude otherwise "[the] requirement for administrative decisions based on substantial evidence and reasoned findings -- which alone make effective judicial review possible -- would become lost in the haze of so-called expertise".

> Accordingly, we cannot accept the Board's unsupported assessment of the prior art.[27]

Accordingly, as the applied art does not disclose all the claimed features, the Examiner has not met his evidentiary burden for anticipation, and reversal of the rejection of claims 7, 11, 12, 18, 22, 23, 29, 33, and 34 by the Honorable Board is requested.

IX. CLAIMS INVOLVED IN THE APPEAL

A copy of the claims involved in the present appeal is attached hereto as Appendix A. As indicated above, the claims in Appendix A do include the amendments filed by Applicant on April 8, 2002.

X. REMARKS CONCERNING NOTICE OF NON-COMPLIANT APPEAL BRIEF

Appellants have submitted this Substitute Appeal Brief to conform to the Examiner's requirements, as set forth in the Notice of Non-Compliance mailed February 19, 2003.

For example, this Substitute Brief eliminates the previously submitted Issue "A", and Arguments associated with the Objection to the Specification. However, as the Board of Appeals and Interferences has stated, "[t]hat part of the Examiner's "objection" which centers on description, enablement and best mode concerns the correspondence of the specification to the statutory requirements set forth in 35 U.S.C. §112 and is within the jurisdiction of this Board."[28]

Appellants disagree with the Examiner's assertion in the Notice of Non-Compliance that the previously submitted Summary of the Invention in the Appeal Brief previously filed December 30, 2002 "...is not reflected in the claims".

However, to further expedite processing of this Appeal, this Substitute Appeal Brief provides specific citation to the Specification at least for each independent claim in each claim group, as suggested by Supervisory Examiner Teska during the February 28, 2003 telephone conference held with the undersigned.

Finally, Appellants have removed the paragraphs objected to in the Summary of the Invention section of the previously submitted Appeal Brief, which were characterized by the Examiner as constituting argument for enablement of the pending claims 1-37.

XI. DISCUSSION OF CONVENTIONAL SOFTWARE ANALYSIS TOOLS

To set the stage for this appeal, and in light of the Examiner's comments and apparent misunderstanding of the novel and non-obvious aspects of the claimed invention, a somewhat longer background discussion of conventionally-available software analysis information is provided in the optionally provided APPENDIX B, as a further aid to the Honorable Board's understanding and appreciation of the claims on appeal, and the context in which the claimed invention was developed.

Respectfully submitted,

By Larry J. Hume

Larry J. Hume
Registration No.: 44,163
CONNOLLY BOVE LODGE & HUTZ, LLP
1990 M Street, N.W., Suite 800
Washington, DC 20036-3425

(202) 331-7111
(202) 293-6229 (Fax)
Attorneys for Applicant

Encl: APP. A – Claims on Appeal
APP. B - Background Discussion of Conventional Software Analysis Information

---

[27] *In re Zurko*, 258 F.3d 1379, 59 U.S.P.Q.2d 1693 (Fed. Cir. 2001).
[28] *Ex parte* C, 27 USPQ 2d 1492, 1494 (B.P.A.I. 1993).

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/241,735

1. (Amended) A software analysis apparatus comprising:

program analysis information generation means for automatically generating program analysis information required for analyzing a computer program;

program analysis information storage means for classifying the program analysis information generated by said program analysis information generation means in an arbitrary unit or at an arbitrary timing, and sequentially storing the program analysis information in a predetermined data recording medium; and

program analysis means for executing program analysis by reading out the program analysis information from said data recording medium.

2. An apparatus according to claim 1, wherein said program analysis means reads out the program analysis information from said data recording medium, and executes program analysis by an interactive process with an operator.

3. An apparatus according to claim 1, wherein said program analysis means reads out the program analysis information from said data recording medium, and executes program analysis by a batch process.

4. An apparatus according to claim 3, wherein said program analysis means generates at least one of metrics information, redundancy information, data flow anomaly information, and maintenance document information by the batch process.

5. An apparatus according to claim 1, wherein said program analysis information generation means generates a plurality of kinds of program analysis information in turn, and said program analysis information storage means stores the program analysis information in said data recording medium every time each kind of program analysis information is generated.

6. An apparatus according to claim 5, further comprising range instruction means for instructing a range of the plurality of kinds of program analysis information to be generated.

7. An apparatus according to claim 1, wherein said program analysis information storage means stores the program analysis information in said data recording medium as a database.

8. An apparatus according to claim 1, wherein the program analysis information includes at least one of:

a syntactic analysis tree generated on the basis of source code of the computer program;

a symbol table indicating meanings of symbols used in source code of the computer program;

a call graph or flow graph generated on the basis of the syntactic analysis tree;

data flow information generated on the basis of the syntactic analysis tree, symbol table, flow graph, and call graph; and

a program dependence graph or module I/O information generated on the basis of the syntactic analysis tree, symbol table, flow graph, call graph, and data flow information.

9. An apparatus according to claim 8, wherein said program analysis information generation means generates the syntactic analysis tree and symbol table, call graph and flow graph, data flow information, and program dependence graph and module I/O information in the order listed.

10. An apparatus according to claim 9, wherein said program analysis information storage means stores each program analysis information in said data recording medium every time said program analysis information generation means generates one of the syntactic analysis tree and symbol table, call graph, flow graph, data flow information, program dependence graph, and module I/O information.

11. (Amended) A software analysis apparatus for generating program analysis information required for analyzing a computer program, comprising:

means for hierarchically registering the generated program analysis information in a database in units of analysis objectives;

means for implementing analysis of the hierarchically registered program analysis information by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program.

12. An apparatus according to claim 11, wherein the database is an object-oriented database.

13. (Amended) A software analysis method, comprising:

automatically generating program analysis information required for analyzing a computer program;

classifying the program analysis information in an arbitrary unit or at an arbitrary timing;

sequentially storing the program analysis information in a predetermined data recording medium; and

executing program analysis by reading out the program analysis information from said data recording medium.

14. A method according to claim 13, wherein the program analysis step includes the step of reading out the program analysis information from said data recording medium, and executing program analysis by an interactive process with an operator.

15. A method according to claim 13, wherein the program analysis step includes the step of reading out the program analysis information from said data recording medium, and executing program analysis by a batch process.

16. A method according to claim 15, wherein the program analysis step includes the step of generating at least one of metrics information, redundancy information, data flow anomaly information, and maintenance document information by the batch process.

17. A method according to claim 13, wherein the program analysis information generation step includes the step of generating a plurality of kinds of program analysis information in turn, and the program analysis information storage step includes the step of storing the program analysis information in said data recording medium every time each kind of program analysis information is generated.

18. A method according to claim 13, wherein the program analysis information storage step includes the step of storing the program analysis information in said data recording medium as a database.

19. A method according to claim 13, wherein the program analysis information includes at least one of:

a syntactic analysis tree generated on the basis of source code of the computer program;

a symbol table indicating meanings of symbols used in source code of the computer program;

a call graph or flow graph generated on the basis of the syntactic analysis tree;

data flow information generated on the basis of the syntactic analysis tree, symbol table, flow graph, and call graph; and

a program dependence graph or module I/O information generated on the basis of the syntactic analysis tree, symbol table, flow graph, call graph, and data flow information.

20. A method according to claim 19, wherein the program analysis information generation step includes the step of generating the syntactic analysis tree and symbol table, call graph and flow graph, data flow information, and program dependence graph and module I/O information in the order listed.

21. A method according to claim 20, wherein the program analysis information storage step includes the step of storing each program analysis information in said data recording medium every time one of the syntactic analysis tree and symbol table, call graph, flow graph, data flow information, program dependence graph, and module I/O information is generated in the program analysis information generation step.

22. (Amended) A software analysis method for generating program analysis information required for analyzing a computer program, comprising:

hierarchically registering the generated program analysis information in a database in units of analysis objectives, and

implementing analysis by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program.

23. (Amended) A method according to claim 22, wherein the database is an object-oriented database.

24. (Amended) A computer readable recording medium recording a computer program for making a computer implement:

a program analysis information generation function of automatically generating program analysis information required for analyzing a computer program;

a program analysis information classification function of classifying the program analysis information generated by the program analysis information generation function in an arbitrary unit or at an arbitrary timing;

a program analysis information storage function of sequentially storing the classified program analysis information generated by the program analysis information classification function in a predetermined data recording medium; and

a program analysis function of executing program analysis by reading out the classified program analysis information from said data recording medium.

25. A medium according to claim 24, wherein the program analysis function reads out the program analysis information from said data recording medium, and executes program analysis by an interactive process with an operator.

26. A medium according to claim 25, wherein the program analysis function reads out the program analysis information from said data recording medium, and executes program analysis by a batch process.

27. A medium according to claim 26, wherein the program analysis function generates at least one of metrics information, redundancy information, data flow anomaly information, and maintenance document information by the batch process.

28. A medium according to claim 24, wherein the program analysis information generation function generates a plurality of kinds of program analysis information in turn, and the program analysis information storage function stores the program analysis information in said data recording medium every time each kind of program analysis information is generated.

29. A medium according to claim 24, wherein the program analysis information storage function stores the program analysis information in said data recording medium as a database.

30. A medium according to claim 24, wherein the program analysis information includes at least one of:

a syntactic analysis tree generated on the basis of source code of the computer program;

a symbol table indicating meanings of symbols used in source code of the computer program;

a call graph or flow graph generated on the basis of the syntactic analysis tree;

data flow information generated on the basis of the syntactic analysis tree, symbol table, flow graph, and call graph; and

a program dependence graph or module I/O information generated on the basis of the syntactic analysis tree, symbol table, flow graph, call graph, and data flow information.

31. A medium according to claim 30, wherein the program analysis information generation function generates the syntactic analysis tree and symbol table, call graph and flow graph, data flow information, and program dependence graph and module I/O information in the order listed.

32. A medium according to claim 31, wherein the program analysis information storage function stores each program analysis information in said data recording medium every time the program analysis information generation function generates one of the syntactic analysis tree and symbol table, call graph, flow graph, data flow information, program dependence graph, and module I/O information.

33. (Amended) A computer readable storage medium recording a program for making a computer implement a function of:

generating program analysis information required for analyzing a computer program,

hierarchically registering the generated program analysis information in a database in units of analysis objectives, and

implementing analysis by reading out the program analysis information already registered in a predetermined layer in correspondence with an analysis objective upon analyzing the computer program.

34. A medium according to claim 33, wherein the database is an object-oriented database.

35. The apparatus of claim 1, wherein said program analysis means analyzes where a given line in a source code of the computer program has an influence on the source code.

36. The method of claim 13, wherein said automatically generating program analysis information includes analyzing where a given line in a source code of the computer program has an influence on an execution of the source code.

37. The computer readable recording medium of claim 24, wherein the program analysis function includes a function of determining where a given line in a source code of the computer program has an influence on an execution of the source code.

APPENDIX B

Background Discussion of Conventional Software Analysis Information

To set the stage for this appeal, and in light of the Examiner's comments and apparent misunderstanding of the novel and non-obvious aspects of the claimed invention, a somewhat longer background discussion of conventionally-available software analysis information is submitted as being necessary to a better appreciation of the claims on appeal.

By way of background, when a large number of programmers develop a large-scale program in collaboration, or maintain a software program already in existence, it is often difficult for a given programmer to understand the program code written by other programmers. The disclosed and claimed invention is directed to not only providing various kinds of program analysis information, such as call graph and flow graph, and automatically analyzing program source code, in order to help understanding of the program, especially for analyzing a large-scale program, but is also directed to a novel and non-obvious way to process the conventionally generated program analysis information.

When a large-scale program is to be analyzed, e.g., millions of lines of code or program steps, the volume of analysis information required for program analysis also becomes very large, oftentimes resulting in poor analysis efficiency. For example, the maximum memory capacity in an analysis computer or workstation may be insufficient for all pieces of analysis information, or the processing time required for analysis may become very long.

When memory capacity is insufficient to accommodate the quantity of program analysis information generated, further analysis of remaining program steps is not possible. In such circumstances, the program to be analyzed is segmented or re-formed into various pieces. However, the program must again be analyzed again from the beginning, resulting in poor analysis efficiency.

If program segmentation is used to facilitate analysis in such a case, certain types of analysis, e.g., influence analysis which detects the influence of a change in value of a given variable in the program on other parts of the program, cannot be relied upon to yield appropriate

and truly representative analysis information which relates to the interactions of the overall program, because of the interruption of the process due to program segmentation.

When the processing time becomes too long, processing may have to be interrupted for higher priority processing tasks, or for other considerations, such as operator fatigue, end of shift, etc. When processing is resumed, the analysis processing in many cases will have to start from the beginning, resulting in inefficiencies, poor analysis efficiency, and further expense.

In addition, when the maximum memory capacity (main storage device used as a work memory) that a normal workstation or personal computer can mount is insufficient for all pieces of analysis information, some serious problems are likely to occur.

For example, when the memory capacity becomes inadequate during analysis, and no more analysis information can be stored in a memory, the remaining analysis for the program cannot be done. In such case, after some modifications are made, e.g., the program to be analyzed is segmented or re-formed into some pieces, the program must be analyzed from the beginning, resulting in poor analysis efficiency.

Furthermore, analysis may be interrupted due to some cause other than memory shortage during program analysis. In such a case, all analysis obtained so far cannot be used, and program analysis must be redone from the beginning, resulting in poor analysis efficiency.

Examples of conventionally available program analysis information are discussed below, and in the Specification at p. 9, line 20, through p. 14, line 20.

A "call graph", for example, indicates a calling relationship among procedures (e.g., "functions" in C) that comprise a program. Nodes of this type of graph represent procedures, and edges represent the calling relationship among the procedures. The call graph shows how much time was spent in each function and its children. From this information, you can find functions that, while they themselves may not have used much time, called other functions that did use unusual amounts of time. When visually displayed, the program analyst or programmer not only can visually recognize the calling relationship among the procedures, but can also easily understand the structured level and contents of a program, or can easily detect an unwanted procedure call.

In contrast, a "flow graph" shows the flow of control in a given procedure. Nodes in this type of graph represent fundamental blocks, and edges represent the flow of control among the fundamental blocks. When this is information is graphically displayed, the control flow in that particular procedure can be visually recognized, and unwanted control flow can be easily detected. For example, "dead code" that control cannot reach is a node which cannot be reached even if the control follows the edges from an entrance node. However, such dead code can easily be detected, since it does not have any link from the entrance node.

A "symbol table" includes a group of information that represent meanings of symbols (variables) used in a program. For example, when the character "a" is used at different locations in a program, they may indicate identical variables or different variables. A symbol table provides a base for the coalescence of data relating to the various elements of the source text and provides a possibility for describing certain relationships between the text and the target machine, such as the assigned, and possibly relative, address of variables. For example, when variables expressed by an identical character are used in two different functions, if those variable are global variables, they are identical variables; and if the variables are locally defined in the individual functions, they represent different variables. A symbol table can provide necessary data for the acquisition of library-provided subprograms, and may also, during run time, act as a transfer vector for the linkage of the generated code and those subprograms.

Also, a "syntactic analysis tree" expresses a program configuration using a tree structure. Upon analyzing a program, syntactic analysis is generally done first by expressing a source program in the form of a tree structure so that the computer can easily understand the program.

The "data flow information" is information resulting from analysis of the data flow such as alias information of a pointer, or definition use chain information that indicates the use location of a variable defined at a given location, for example. Using this data flow information, any data flow anomalies can be inspected.

A "data flow anomaly" is a combination of illegal events with respect to data. All data must be used while observing specified rules, i.e., each data must be used after it is defined, and may not be used after it is finally undefined. An illegal combination that does not abide by such rules causes a data flow anomaly. Even when data flow anomaly is present, source code can be

compiled. However, upon executing the compiled program in practice, a data flow anomaly may appear when control takes an anomalous path. Hence, such an anomaly is preferably removed in advance in the processing of source code.

A "program dependence graph" expresses control dependence in a program (dependence among functions), and data dependence (when data are substituted in functions) in the form of a graph. When the program dependence graph is generated, for example, influence range analysis of a program can be executed. The influence range analysis analyzes the range of influence imposed when a certain location (one sentence or one variable, for example) in a program has been changed.

"Module I/O information" pertains to input/output ("I/O") of modules (e.g., functions in C) that form a program. This information describes I/O information using I/O functions and global variables, in addition to I/Os as arguments and return values.

"Metrics information" pertains to numeration indices of software. Metrics which represent quantitative complexity, and those which represent qualitative complexity may be generated. As metrics of quantitative complexity, two different kinds of quantities, i.e., a size metric that measures the physical description quantity of a program, and cyclomatic number that measures the complexity of a control structure are measured. On the other hand, as metrics of qualitative complexity, cohesion and coupling that express the contents of modules are often measured.

"Redundancy information" pertains to a redundant sentence that does not influence output in a single procedure. Since a redundant sentence does not influence output, if it is deleted from the source code, the external output remains unchanged. Hence, by confirming such redundant sentences in accordance with this redundancy information, and taking measures against them, unintended operation can be prevented.

"Maintenance document information" refers to a document group used upon maintaining a program. More specifically, this information describes lists of procedures, types, variable names defined in a program, for example, and how the individual procedures are related. This "maintenance document information" may be conventionally used when a given programmer

wants to understand a program created by another programmer. By acquiring the "maintenance document information", information can be provided in the hypertext format, and is far more convenient as compared to paper-based documents.

These are non-exhaustive examples of conventional program analysis information commonly used and understood in a software development or software maintenance environment.

What conventionally known software program analysis devices and techniques lack is a way to handle such types of conventionally obtained analysis information generated from large scale programs, without making large demands upon memory, taking longer than desired to process, or requiring program segmentation which may adversely impact the results of analysis information obtained from overall program interactions, and which consequently do not accurately represent the true program interactions. What conventionally known software program analysis devices and techniques also lack is a way to compensate for a failure of a previous process to complete, without having to start the analysis completely over.